The Tableau WorkBench

Pietro Abate

Research School of Information Science and Engineering
Australian National University

December 4, 2006
Outline

Motivations

Introduction

Mechanizing Tableau Methods

Using the TWB

Case Studies

Unified Branching Logic (UB)

Final Remarks
Motivations

Introduction

Mechanizing Tableau Methods

Using the TWB

Case Studies

Unified Branching Logic (UB)

Final Remarks
TWB At First Glance

- It is generic and extensible
- Modular framework to build theorem provers
- It is based on a general tableau algorithm
- Front-end for tableau calculi

Motto : Simple things should be simple, difficult things should be possible.
State of the Art

- **Direct Provers**
  - Fact/Fact++ (Description logics)
  - LWB (Modal logics, intuitionistic logics)
  - KSAT (modal logic)
  - Lotrec (arbitrary logics)

- **Translational Provers**
  - SPASS / MSPASS (first order logics, modal logics)
  - Vampire (first order logic)
Target Users

- Logicians without programming/technical background:
  - to experiment with new logics
  - to prototype theorem provers
  - to compare different technique using the same framework

- Automated reasoning teachers:
  - syntax similar to textbooks
  - no need to learn a programming language (for simple tasks)

- Expert automated reasoner with programming skills:
  - can be used as an embedded prover
  - the user level can be bypassed and each prover can be highly optimized
Outline

Motivations

Introduction

Mechanizing Tableau Methods

Using the TWB

Case Studies

Unified Branching Logic (UB)

Final Remarks
Tableau Methods

- Traditional tableau methods:
  - Modular
  - Efficient
  - Simple

  Traditional tableau calculi specify what it can be done, not how to do it.

- Extensions to the traditional formulation:
  - Histories (termination)
  - Variables
  - Tactic language
Outline

Motivations

Introduction

Mechanizing Tableau Methods

Using the TWB

Case Studies

Unified Branching Logic (UB)

Final Remarks
Definitions

A tableau rule (name) \( \frac{n}{d_1 \ldots d_n} \) (side cond) is defined as:

- numerator, denominators
- principal formula, side formula
- each \( n, d_1 \ldots d_n \) is a list of formula schema, e.g. \( A \& B; X \), where \( A, B \) and \( X \) are meta variables
- side conditions (list of boolean functions)

A node is composed of a set of formula plus other data structures (Histories and Variables). A tableau for a set \( X \) is a tree of nodes with root \( X \) where the children of a node is obtained by instantiating a tableau rule.

- A node is partitioned according to a rule numerator
- A rule is applicable to a node if there exists a partition of the node and all side conditions are true.
A branch in a tableau is a linear sequence of nodes.

A tableau is open if there exists a branch that is open.

The tableau method search for the existence of an open branch.
Rule Classifications

- **Terminal Rule**: \((\text{name}) \frac{n}{\bot}\)

- **Linear Rule**: \((\text{name}) \frac{n}{d}\)

- **Universally Branching Rule**: \((\text{name}) \frac{n}{d_1 | d_2}\)

- **Existentially Branching Rule**: \((\text{name}) \frac{n}{d_1 || d_2}\)

- **Conditionally Branching Rule**: \((\text{name}) \frac{n}{d_1 ||| d_2}\)
Removing Non-Determinism

- Node Choice: Out of many, we select a node to expand (the current node)
- Rule Choice: Given a node, we select a rule that is applicable to the current node
- Formula Choice: Given a node and a rule, we select which formula is the principal formula
### How big?

Core library by lines of code (loc)

<table>
<thead>
<tr>
<th>component</th>
<th>loc</th>
</tr>
</thead>
<tbody>
<tr>
<td>core</td>
<td>464</td>
</tr>
<tr>
<td>tableau library</td>
<td>818</td>
</tr>
<tr>
<td>syntax library</td>
<td>1624</td>
</tr>
<tr>
<td>data-type library</td>
<td>664</td>
</tr>
<tr>
<td>application (cli)</td>
<td>226</td>
</tr>
<tr>
<td><strong>total</strong></td>
<td><strong>3800</strong></td>
</tr>
</tbody>
</table>
# How small?

Logic modules by lines of code (loc)

<table>
<thead>
<tr>
<th>logic</th>
<th>loc (tableau)</th>
<th>loc (functions)</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPL</td>
<td>18</td>
<td>98</td>
<td>126</td>
</tr>
<tr>
<td>K</td>
<td>21</td>
<td>72</td>
<td>93</td>
</tr>
<tr>
<td>KT</td>
<td>31</td>
<td>72</td>
<td>103</td>
</tr>
<tr>
<td>S4</td>
<td>41</td>
<td>72</td>
<td>113</td>
</tr>
<tr>
<td>PLTL</td>
<td>148</td>
<td>194</td>
<td>242</td>
</tr>
</tbody>
</table>
Outline

Motivations

Introduction

Mechanizing Tableau Methods

Using the TWB

Case Studies

Unified Branching Logic (UB)

Final Remarks
Tableau calculus for K

\[
(\bot) \quad \frac{\varphi; \neg \varphi; \Lambda}{\bot} \\
(\wedge) \quad \frac{\varphi \wedge \psi; \Gamma}{\varphi; \psi; \Gamma} \\
(\vee) \quad \frac{\varphi \vee \psi; \Gamma}{\varphi; \Gamma | \psi; \Gamma} \\
(K) \quad \frac{\Diamond \varphi; \Box \Sigma; \Lambda}{\varphi; \Sigma}
\]
Tableau calculus for S4 with histories

\[\begin{align*}
\text{(⊥)} & \quad \frac{\varphi; \neg \varphi; \Lambda}{\bot} :: \ldots \quad \text{(∧)} & \quad \frac{\varphi \land \psi; \Gamma}{\varphi; \psi; \Gamma} :: \ldots \\
\text{(∨)} & \quad \frac{\varphi \lor \psi; \Gamma}{\varphi; \Gamma :: \ldots | \psi; \Gamma :: \ldots} \\
\text{(T)} & \quad \frac{\Box \varphi; \Gamma :: \Pi, \Sigma}{\varphi; \Gamma :: \emptyset, \{\varphi\} \cup \Sigma} \quad \varphi \notin \Sigma \\
\text{(S4)} & \quad \frac{\Diamond \varphi; \Lambda :: \Pi, \Sigma}{\varphi; \Sigma :: \{\Diamond \varphi\} \cup \Pi, \Sigma} \quad \Diamond \varphi \notin \Pi
\end{align*}\]
Rule Syntax (1/4)

CONNECTIVES

And, "__&__", Two;
Or, "__v__", Two;
Imp, "__->__", One;
DImp, "__<-->__", One;
Not, "~__", Zero;
Dia, "Dia__", Zero;
Box, "Box__", Zero

END

HISTORIES

(DIAMONDS : Set of Formula := new Set.set);
(BOXES : Set of Formula := new Set.set)

END
Rule Syntax (2/4)

RULE Id
{ a } ; { ~ a } ; x
END

RULE And
{ a & b } ; x
END

(⊥) \[ \frac{\varphi; \neg\varphi; \Lambda}{\bot} \]

(∧) \[ \frac{\varphi \land \psi; \Gamma}{\varphi; \psi; \Gamma} \]
Rule Syntax (3/4)

RULE Or
\{ a \lor b \} ; x

\[ (\lor) \frac{\varphi \lor \psi; \Gamma}{\varphi; \Gamma \mid \psi; \Gamma} \]

a ; x \mid b ; x

END

RULE K
\{ \text{Dia} \ a \} ; \text{Box} \ x ; z

\[ (K) \frac{\Diamond \varphi ; \Box \Sigma ; \Lambda}{\varphi ; \Sigma} \]

a ; x

END
Rule Syntax (4/4)

RULE S4
{ Dia a } ; z
----------------------
a ; SIGMA

COND notin(Dia a, PI)
ACTION [ PI := add(Dia a, PI) ]
END

\[(S4)\]
\[
\begin{align*}
\diamond \varphi; \Lambda &:: \Pi, \Sigma \\
\varphi; \Sigma &:: \{\diamond \varphi\} \cup \Pi, \Sigma \\
\end{align*}
\]
Strategy

- $t = \text{Skip} : t$ always succeeds
- $t = \text{Fail} : t$ always fails
- $t = \text{Rule}(r) : t$ succeeds when rule $r$ is applicable.
- $t = t_1; t_2 : t$ fails if either $t_1$ or $t_2$ fail, succeeds otherwise.
- $t = t_1 | t_2 : t$ fails if both $t_1$ and $t_2$ fail, succeeds otherwise.
- $\text{Repeat}(t) = \mu(X).(t; X|\text{Skip}) :$ If tactic $t$ succeeds, the tactic $\text{Repeat}(t)$ behaves like $t; \text{Repeat}(t)$. If $t$ fails, then the tactic $\text{Repeat}(t)$ behaves like $\text{Skip}$. Repeat always succeeds.
Strategies for K and S4

- $K : \text{Repeat(Repeat(And|Or|Id); K)}$
- $S4 : \text{Repeat(Repeat(And|Or|T|Id); S4)}$
Outline

Motivations

Introduction

Mechanizing Tableau Methods

Using the TWB

Case Studies

Unified Branching Logic (UB)

Final Remarks
Optimizations : Simplification

RULE And
\{ a \& b \} ; x

a[b]; b[a] ; x[a][b]
END

RULE Or
\{ a \lor b \} ; x

a ; x[a] | b ; x[b]
END
Optimizations : Simplification

let rec simpl phi a =
let rec aux phi a = match a with
| term (~ b) when b = a -> term(Falsum)
| term (~ b) -> term ( ~ [aux phi b] )
| term (b & c) ->
    term ( [aux phi b] & [aux phi c] )
| term (b v c) ->
    term ( [aux phi b] v [aux phi c] )
| _ when phi = a -> term(Verum)
| _ when phi = (nnf (term ( ~ a ))) ->
    term(Falsum)
| _ -> a
in boolean (aux phi a)
Optimizations : Simplification

\[\neg \varphi \lor \varphi \rightarrow T \quad \varphi \lor T \rightarrow T \quad \varphi \lor \bot \rightarrow \varphi \quad \varphi \lor \varphi \rightarrow \varphi\]

\[\neg \varphi \land \varphi \rightarrow \bot \quad \varphi \land T \rightarrow \varphi \quad \varphi \land \bot \rightarrow \bot \quad \varphi \land \varphi \rightarrow \varphi\]

\[\neg \bot \rightarrow T \quad \neg T \rightarrow \bot\]
Benchmarks

\( s4 \) is the naive tableau prover for \( S4 \). The only optimization is caching;

\( s4c \) is as \( s4 \) but without using caching;

\( s4s \) is as \( s4 \) but using caching and simplification and semantic branching;

\( s4bj \) is as \( s4 \) but using caching and back-jumping and semantic branching;

\( s4sbj \) uses all the previously mentioned optimization.
Benchmarks

Provable formulae

Non-Provable formulae

Table: S4 (time)
Benchmarks

![Graph showing benchmarks for provable and non-provable formulae.](image)

**Provable formulae**

**Non-Provable formulae**

**Table:** S4 (rules)
Outline

Motivations

Introduction

Mechanizing Tableau Methods

Using the TWB

Case Studies

Unified Branching Logic (UB)

Final Remarks
Unified Branching Logic (UB)

- Introduced by Ben-Ari et al. in 1982
- Subsumed by CTL (Emerson et al. 1982)
- It extends linear time logic without considering the operator Until.
Syntax

\[ p ::= p_1 \mid p_2 \mid \cdots \]
\[ \varphi ::= p \mid \neg \varphi \mid \varphi_1 \lor \varphi_2 \mid \varphi_1 \land \varphi_2 \mid \bot \mid \top \]
\[ \mid AG \varphi \mid AF \varphi \mid AX \varphi \]
\[ \mid EG \varphi \mid EF \varphi \mid EX \varphi \]
Semantics

Let $M = (S, R, L)$ be a UB-model. The truth value $M, s \vDash \varphi$ of a formula $\varphi$ in a state $s \in S$ is recursively defined as follows:

1. $M, s \vDash p$ iff $p \in L(s)$
2. $M, s \vDash \neg \varphi$ iff $M, s \nvDash \varphi$
3. $M, s \vDash \varphi_1 \lor \varphi_2$ iff $M, s \vDash \varphi_1$ or $M, s \vDash \varphi_2$
4. $M, s \vDash \varphi_1 \land \varphi_2$ iff $M, s \vDash \varphi_1$ and $M, s \vDash \varphi_2$
5. $M, s \vDash EX\varphi$ iff $\exists t \in S$ such that $sRt$ and $M, t \vDash \varphi$
6. $M, s \vDash AF\varphi$ iff $\forall$ fullpaths $b$ with $b_0 = s$, $\exists t \in b$ such that $M, t \vDash \varphi$
7. $M, s \vDash EF\varphi$ iff $\exists$ a fullpath $b$ with $b_0 = s$ and $\exists t \in b$ such that $M, t \vDash \varphi$

The semantic of $AX$, $AG$ and $EG$ is given by duality with $EX$, $EF$ and $AF$, respectively.
Intuition for AG
Intuition for EG
Intuition for AF
Intuition for EF
Linear Rules

\[(\land) \quad \frac{\varphi \land \psi; \Gamma \vdash \cdots}{\varphi; \psi; \Gamma \vdash \cdots} \quad (D) \quad \frac{\Gamma \vdash \cdots}{\text{EX} \top; \Gamma \vdash \cdots} \quad \text{EX} \varphi \notin \Gamma\]

\[(AG) \quad \frac{\text{AG} \varphi; \Gamma \vdash \cdots}{\varphi; \text{AXAG} \varphi; \Gamma \vdash \cdots} \quad (EG) \quad \frac{\text{EG} \varphi; \Gamma \vdash \cdots}{\varphi; \text{EXEG} \varphi; \Gamma \vdash \cdots}\]

- The \((D)\) enforces seriality
- The \((AG)\) captures the axioms \(\text{AG} \varphi \leftrightarrow \varphi \land \text{AXAG} \varphi\)
- The \((EG)\) captures the axioms \(\text{EG} \varphi \leftrightarrow \varphi \land \text{EXEG} \varphi\)
Universally Branching Rules

\[

(EF) \quad EF\varphi \ ; \ \Gamma :: \ Fev, \ Br :: \ uev \\
\varphi \ ; \ \Gamma :: \ \{EF\varphi\} \cup Fev, \ Br :: \ uev_1 \ | \ EXEF\varphi \ ; \ \Gamma :: \ Fev, \ Br :: \ uev_2
\]

\[

(AF) \quad AF\varphi \ ; \ \Gamma :: \ Fev, \ Br :: \ uev \\
\varphi \ ; \ \Gamma :: \ \{AF\varphi\} \cup Fev, \ Br :: \ uev_1 \ | \ AXAF\varphi \ ; \ \Gamma :: \ Fev, \ Br :: \ uev_2
\]

\[

(\lor) \quad \varphi \lor \psi \ ; \ \Gamma :: \ \cdots :: \ uev \\
\varphi \ ; \ \Gamma :: \ \cdots :: \ uev_1 \ | \ \psi \ ; \ \Gamma :: \ \cdots :: \ uev_2
\]

- The \((EF)\) captures the fix-point \(EF\varphi \Leftrightarrow \varphi \lor EXEF\varphi\)
- The \((EF)\) captures the fix-point \(AF\varphi \Leftrightarrow \varphi \lor AXAF\varphi\)
- The \((\lor)\) rule is standard except for computation of \(uev\)
Universally Branching Rules Conditions

where in the \((EF)\), \((AF)\) and \((\lor)\) rules:

\[
f(x, y) = \{ i \mid (x, i) \in y \}
\]

\[
UEF = \{ (EF\psi, n) \mid f(EF\psi, uev_1) \neq \emptyset \& f(EF\psi, uev_2) \neq \emptyset \\
\& n = \min (f(EF\psi, uev_1) \cup f(EF\psi, uev_2)) \}
\]

\[
UAF = \{ (AF\psi, n) \mid f(AF\psi, uev_1) \neq \emptyset \& f(AF\psi, uev_2) \neq \emptyset \\
\& n = \max (f(AF\psi, uev_1) \cup f(AF\psi, uev_2)) \}
\]

\[
uev := \begin{cases} 
uev_1 & \text{if } uev_2 = \{ (false, _) \} \\
uev_2 & \text{if } uev_1 = \{ (false, _) \} \\
UEF \cup UAF & \text{otherwise}
\end{cases}
\]
Unfolding the fix-point

$\bot \leftarrow EF\varphi$

$EF\varphi \leftarrow \bot$

$EF\varphi \leftarrow \bot$

$EF\varphi \leftarrow \bot$

$EF\varphi \leftarrow \bot$

$EF\varphi$ loop target

$EF\varphi$ loop

$EF\varphi$ loop source
Unfolding multiple fix-points

\[ EF \varphi; \]
\[ EF \psi \]

\[ \varphi; \]
\[ EXEF \psi \]

\[ \varphi; \]
\[ EXEF \psi \]

\[ EXEF \varphi; \]
\[ \psi \]

\[ EXEF \varphi; \]
\[ \psi \]

\[ EXEF \varphi; \]
\[ EXEF \psi \]
Intuition for the Universally Branching Rules Conditions

The criteria for computing $uev$ are as follows:

- if either branch is “closed” then keep the other $uev$. If both are “closed” then the $uev = uev_1 = \{(\text{false}, \_}\}$.
- if either $uev_1$ or $uev_2$ is empty because it has no “procrastinators”, then the numerator is empty because $UEF = UAF = \emptyset$, hence $UEF \cup UAF = \emptyset$.
- if both are not “closed” and both contain “procrastinators”, then $UEF \cup UAF$ constructs their “intersection” (sic!).

The crucial point is to simply ignore the notion of “open” and “fulfilled” and consider only “procrastinators”.
Existentially Branching Rules

\[
\begin{aligned}
(EX) & \quad EX \varphi_1 ; \ldots ; EX \varphi_n ; EX \Gamma ; AX \Delta ; \land \\
& \quad :: Fev, Br :: uev \\
& \quad \varphi_1 ; \Delta \\
& \quad :: \emptyset, Br_1 :: uev_1 \parallel \cdots \parallel :: \emptyset, Br_n :: uev_n \parallel :: Fev, Br :: uev_{n+1} \\
& \quad \hat{(1)} \parallel \cdots \parallel \hat{(n)} \parallel \hat{} \\
\end{aligned}
\]
Existentially Branching Rules Conditions

with:

\[ n \geq 1 \]

\[ \ddagger(i) \text{ is the condition } \forall j . 0 \leq j \leq \text{len}(Br) \Rightarrow \{ \varphi_i \} \cup \Delta \neq Br[j].\text{core} \]

\[ \dagger \text{ is the (blocking) condition } \forall \psi \in \Gamma . \exists j \geq 0 \text{ s.t. } \{ \psi \} \cup \Delta = Br[j].\text{core} \]

and where for \( 1 \leq i \leq n \):

\[ Br_i = (\{ \varphi_i \} \cup \Delta, \text{Fev}).Br \]

\[ l = \text{len}(Br) - 1 \]

\[ UEV = \bigcup_{j=1}^{n+1} uev_j \]

\[ uev := \begin{cases} UEV & \text{if } (\text{false, } _) \notin UEV \& \forall (_, n) \in UEV . n \leq l \\ \{ (\text{false, } l) \} & \text{otherwise} \end{cases} \]
Intuition for the Existentially Branching Rule Conditions

The criteria for computing $uev$ are as follows:

- if the $i^{th}$ branch is closed than $uev = \{ (\_\_\_, \text{false}) \}$.
- if any $uev$ contains an iterated eventuality that loops lower then this point ($n \leq \text{len}(Br)$) then $uev = \{ (\_\_\_, \text{false}) \}$.
- if all eventualities loop higher than this point, then $uev$ is equal to the union of all $uevs$. 
Intuition for the Existentially Branching Rule

\[
\begin{align*}
EX \varphi_1 ; \ldots ; EX \varphi_n ; \\
EX \Gamma ; AX \Delta ; \land
\end{align*}
\]

\[
\begin{align*}
\varphi_1 ; \Delta \\
\varphi_n ; \Delta
\end{align*}
\]
Terminal Rules

\[(id) \quad \frac{EX\Gamma ; AX\Delta ; \land :: Fev, Br :: uev}{Stop} \quad \{\neg p; p\} \subseteq \land\]

where \( uev := \{(false, len(Br))\}\)

\[(block) \quad \frac{EX\Gamma ; AX\Delta ; \land :: Fev, Br :: uev}{Stop} \quad \{\neg p; p\} \not\subseteq \land \text{ and } \dagger\]
# Block Rule Conditions

where † is the condition \( \forall \psi \in \Gamma \cdot \exists j \geq 0 \cdot \{ \psi \} \cup \Delta = Br[j].\text{core} \), and

\[
\begin{align*}
\text{Cores} & = \{ \{ EF\varphi \} \cup \Delta \mid EF\varphi \in \Gamma \} \\
\text{UAF} & = \{ (AF\varphi, i) \mid \exists c \in \text{Cores}, \exists i \text{ such that } c = Br[i].\text{core} \\
& \quad \& \forall j . i \leq j \leq \text{len}(Br) \Rightarrow AF\varphi \in Br[j].\text{core} \\
& \quad \& AF\varphi \notin Br[j].\text{fev} \} \\
\text{UEF} & = \{ (EF\varphi, i) \mid \exists i \text{ such that } \{ EF\varphi \} \cup \Delta = Br[i].\text{core} \\
& \quad \& \forall j . i \leq j \leq \text{len}(Br) \Rightarrow EF\varphi \in Br[j].\text{core} \\
& \quad \& EF\varphi \notin Br[j].\text{fev} \} \\
uev & := \{ (\psi, n) \in \text{UAF} \cup \text{UEF} \mid \psi \notin \text{Fev} \}
\end{align*}
\]
Intuition for the Block Rule Conditions

A blocked iterated eventuality \((EV\varphi, n)\) is in \(uev\) iff

- The branch \(b\) from the loop target to the loop source is a procrastinator for \(EV\varphi\).
- \(EV\varphi\) is not fulfilled on \(b\) between the target and the source of the loop.
Soundness and Completeness

Proposition (Termination)

All UB-tableau for a node $\varphi :: \text{Fev}, \text{Br} :: \text{uev}$ always terminate.

Theorem

If $\mathcal{T}$ is an expanded tableau for $\varphi :: \text{Fev}, \text{Br} :: \text{uev}$, with $\text{uev} = \emptyset$ then $\varphi$ is UB-satisfiable.

Theorem

If a formula $\varphi$ is UB-satisfiable then there exists an expanded tableau for $\varphi :: \cdots :: \text{uev}$ where $\text{uev} = \emptyset$. 
Outline

Motivations

Introduction

Mechanizing Tableau Methods

Using the TWB

Case Studies

Unified Branching Logic (UB)

Final Remarks
Conclusions

- The TWB is small compared with other theorem provers;
- It is generic and extendible;
- It can be used to experiment with new logics, but in principle also to build specialized TPs;
- It can be extended to incorporate well known optimizations.
Availability

- One The Web:
  http://twb.rsise.anu.edu.au

- Demo:
  http://twb.rsise.anu.edu.au/twbdemo

- Download (Via Darcs):
  http://twb.rsise.anu.edu.au/Repository